(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) EP 2 293 488 A1

(12) EUROPEAN PATENT APPLICATION

(43) Date of publication:
09.03.2011 Bulletin 2011/10

(51) Int Cl.:
H04L 9/06 *(2006.01)*

(21) Application number: **10165439.0**

(22) Date of filing: **09.06.2010**

(84) Designated Contracting States:
AL AT BE BG CH CY CZ DE DK EE ES FI FR GB GR HR HU IE IS IT LI LT LU LV MC MK MT NL NO PL PT RO SE SI SK SM TR
Designated Extension States:
BA ME RS

(30) Priority: **04.09.2009 FR 0956053**

(71) Applicant: **Oberthur Technologies**
92300 Levallois-Perret (FR)

(72) Inventors:
- **Prouff, Emmanuel**
  75018 Paris (FR)
- **McEvoy, Robert**
  Cork (IE)

(74) Representative: **Le Tourneau, Augustin et al**
Santarelli
14, avenue de la Grande Armée
B. P. 237
F-75822 Paris Cedex 17 (FR)

(54) Method for cryptographic processing of data units

(57) The present invention concerns a method for cryptographic processing of secret data units. This method includes:
- a permutation step during which elementary data units of each data unit are permutated, and
- at least one cryptographic operation applied to the permutated data units.

According to the invention, prior to said cryptographic operation step, a different permutation is applied to each of the said data units.

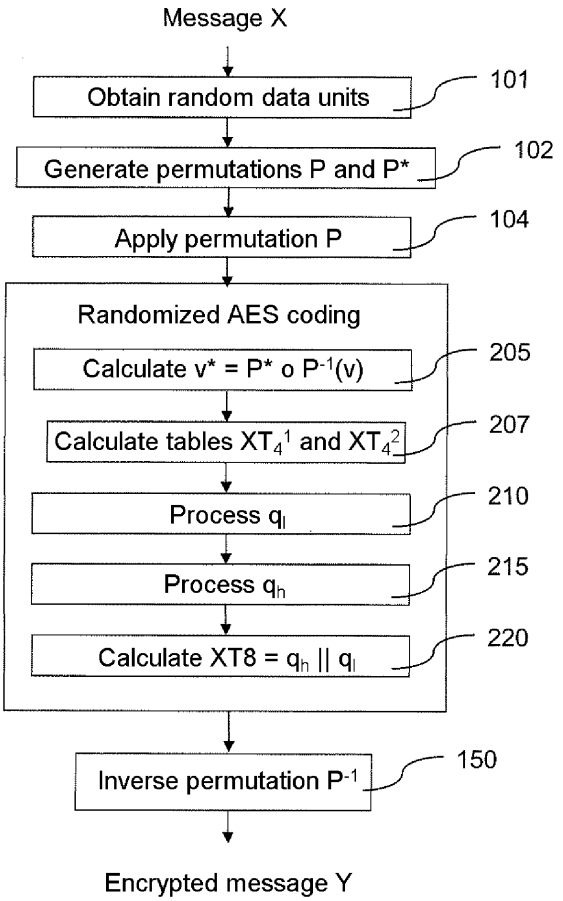

Figure 4

EP 2 293 488 A1

Description

[0001] The present invention concerns a method for cryptographic processing of data units, in particular to protect them against side channel analysis (SCA) attacks, in particular differential power analysis (DPA) attacks, and an associated device. It applies in particular to protecting contents to be kept secret or electronic circuits and microcircuit cards.

[0002] In some data processing methods, in particular in the context of cryptographic data processing, the processing algorithms use data that must remain secret (for example cryptographic keys) to ensure that the system operates with the required security. These types of methods are the target for attacks by malicious users seeking to outsmart the security features of the system.

[0003] When a cryptographic algorithm is implemented in hardware or in an embedded program, information relating to intermediate variables can "leak" during processing. This information can be present in electrical power consumption, electromagnetic emissions or time characteristics. The SCA class of attacks attempts to exploit these leaks to retrieve secret information (generally a secret key or key bits).

[0004] The DPA attack obtains information on the secret key (contained in a smart card, for example) by performing a statistical analysis of records of electrical power consumption measured over a large number of calculations using the same key. This type of attack does not require any knowledge of the individual electrical power consumption of each instruction or the position in time of each instruction. It is applied in exactly the same way once the attacker knows the outputs of the algorithm and the corresponding power consumption curves.

[0005] A known generic countermeasure separates all the intermediate variables (Suresh Chari, Charantjit S. Jutla, Josyula R. Rao and Pankaj Rohatgi, "Towards Sound Approaches to Counteract Power-Analysis Attacks", Proceedings of Advances in Cryptology - CRYPTO'99, Springer-Verlag, 1999, pages 398-412). This general method greatly increases the resources necessary in terms of memory space and/or calculation time. Moreover, the intermediate steps can also suffer DPA attacks, and the variables must therefore be separated over all steps of the algorithm. This exacerbates the problem of additional resource consumption, in particular for embedded systems such as smart cards.

[0006] A general countermeasure masks all inputs and outputs of each elementary operation executed by the microprocessor, i.e. combines them with a mask m. To protect against side channel analysis (SCA) attacks and in particular differential power analysis (DPA) attacks, it is known to use additive masking whose equation is

X XOR m

where X is the Galois field $GF(2^8)^{16}$,
XOR is the exclusive-OR function that associates with a pair (0,0) the value 0, with the pair (0,1) the value 1, with the pair (1,0) the value 1 and with the pair (1,1) the value 0, and
m is a random number.

[0007] Note that when a bit of m has the value 1 the value of the corresponding bit of X is inverted on masking and when a bit of m has the value 0 the value of the corresponding bit of X remains the same on masking. This amounts to stating that (X XOR m) XOR m = X. Thus demasking is effected by the same operation as masking.

[0008] If the source of random numbers is uniformly distributed, the data resulting from masking is likewise and no longer depends on the secret key.

[0009] The paper by J.-S. Coron "A new DPA countermeasure based on permutation tables" in "Security and cryptography for networks", 6th international conference SCN 2008, volume 5229 in "Lecture Notes in Computer Science", pages 278, 292, Springer 2008, proposes a countermeasure using permutation tables. This proposal can be seen as a generalization of the standard approach, masking no longer being effected by random translation but by random permutation. As in standard masking, the countermeasure using permutation tables necessitates a random data string, which is used at the beginning of the algorithm to generate a permutation P. In the case of the encryption algorithm, the permutation P is applied both to the message X to be encrypted and to the secret key K, to produce P(X) and P(K). The encryption algorithm uses these permutated variables. In each step of the encryption algorithm, the cryptographic operations must be modified so that all the intermediate variables remain in the permutated form defined by the permutation P.

[0010] In the AES context, the intermediate variables are words of eight bits. The permutation P includes two permutations of four bits, P_1 and P_2, and is applied to a variable X of eight bits, or bytes, according to the following formula, in which X_h and X_l respectively represent the MSB half-byte and the LSB half-byte of the byte X and "||" represents concatenation:

$$P(X) = P_2(X_h) \,||\, P_1(X_l).$$

[0011] Each time the algorithm is called, the permutations P_1 and P_2 are chosen randomly in a set of permutations defined on F_2^4, set of four bits, also denoted by $GF(2)^4$. In the remainder of the description, it is assumed that the random variable P_1 (respectively P_2) associated with the random generation of p_1 (respectively p_2) satisfies the condition whereby the probability that the event $P_1 = p_1$ is equal to 2^{-4d} (respectively the probability that $P_2 = p_2$ is equal to 2^{-4d}), d being an integer (typically d = 4), for each value of p_1 (respectively p_2).

[0012] The AES standard is a well-known block encryption algorithm. The AES round function for encryption operates on elements of 16 bytes a_i with i running from 0 to 15, and consists of four transformations: AddRoundKey, SubBytes, ShiftRows and MixColumns.

[0013] Hereinafter, the use of permutations in the AES context is referred to as "randomized AES", the term "randomized" added at the start of an operation or transformation indicating that it operates on permutated data. After the random permutation has been generated, it is applied to each byte of the message and the key. For each byte x, u = P(x) denotes the representation of x by the permutation P.

[0014] Each permutated value is processed by the AES round function, in which it is subjected to the transformation indicated above. Each of the AES transformations must be scrupulously implemented so that:

- the sensitive variables always appear in their form permutated by P, and
- the output of each transformation is a permutated form.

[0015] Implementations of AddRoundKey and MixColumns are given below:

- randomized AddRoundKey takes two bytes u = P(x) and v = P(y) as input and supplies as output P(x XOR y) where XOR represents the exclusive-OR operation. To this end, two tables with input on 8 bits and output on 4 bits are defined for (u_l, v_l) and (u_h, v_h) in $(F_2^4)^2$:

$$XT_4^1(u_l \,||\, v_l) = p_1(p_1^{-1}(u_l) \text{ XOR } p_1^{-1}(v_l))$$

$$XT_4^2(u_h \,||\, v_h) = p_2(p_2^{-1}(u_h) \text{ XOR } p_2^{-1}(v_h))$$

where p_1 and p_2 are such that $P(X) = p_1(x_l)\|p_2(x_h)$ and "p_x^{-1}" denotes the inverse permutation of permutation "p_x".

[0016] The tables XT_4^1 and XT_4^2 are calculated at the same time as P and stored in memory. An XOR function on eight bits XT_8 is then calculated using look-up tables (LUT) for u and v in F_2^8.

$$XT_8(u,v) = XT_4^2(u_h \,||\, v_h) \,||\, XT_4^1(u_l \,||\, v_l).$$

- Randomized MixColumns is calculated as a combination of multiplication-by-two operations, and XOR operations. To calculate randomized MixColumns from $P(a_i)$, the representation of the bytes as permutated by P, the XOR operations are calculated using the function XT_8. For the multiplication-by-two operation, a function D_2 is defined such that, when it is applied to u = P(x), $D_2(P(x)) = P(<'02\} \,.\, x)$ is obtained, where {.} represents hexadecimal notation and "." represents multiplication modulo $x^4 + 1$. The permutated representation of the first byte of the output of MixColumns is calculated as follows:

  $P(a_0^{new}) = XT_8\,(D_2(a'_0), XT_8(D_2(a'_1), XT_8(a'_1, XT_8(a'_2, a'_3))))$ in which formula a'_i represents the representation of a_i permutated by P. The other bytes in the outputs of randomized MixColumns can be calculated similarly, using XT_8 and D_2.

[0017] At the end of the last round, the inverse permutation P^{-1} is applied to each byte of the current result, to reveal the encrypted message.

[0018] The inventors have determined that this randomized AES is vulnerable, in particular because two identical data units produce two identical permutated data units. The present invention aims to remedy this drawback.

[0019] To this end, according to a first aspect, the invention provides a method for cryptographic processing of data units, that includes:

- a permutation step during which elementary data units of each data unit are permutated, and

- at least one cryptographic operation applied to the permutated data units,

characterized in that, prior to said cryptographic operation step, a different permutation is applied to each of the said data units.

[0020] Thus two identical data units are rendered different by the permutation step, which reduces the level of vulnerability referred to above. According to particular features, the method of the present invention further includes a step of decomposition of each of said data units into data unit parts, each data unit part including a plurality of elementary data units, and, during the permutation step, different permutations are effected for the same parts of said data units.

[0021] The quantity of memory necessary for processing the data units is much smaller with separate processing of the data unit parts than if the data unit itself were processed because of the routine use of look-up tables to implement operations. For example, processing two bytes would necessitate a look-up table with 2^{16} binary inputs, which is not compatible with the quantities of memory available, in particular in portable electronic units.

[0022] According to particular features, the permutation step includes:

- a first permutation step during which the same permutation is applied to the different data units, and
- a second permutation step during which different permutations are applied to the different data units.

[0023] According to particular features, the permutation step includes:

- a first step of permutation of the data units during which a first permutation is applied to a first part of each of the data units and a second permutation, different from the first permutation, is applied to a second part of each of the data units, the first and second permutations being different, and
- a second permutation step during which the second permutation is applied to the first part of one of the data units after the inverse permutation of the first permutation, and the first permutation is applied to the second part of said data units after the inverse permutation of the second permutation.

[0024] Thus only a small number of permutations and look-up tables are used, which reduces the processing time and memory space resources necessary for implementing the present invention.

[0025] According to particular features, said permutations are statistically independent.

[0026] These features eliminate the level 1 vulnerabilities referred to above.

[0027] According to particular features, at least one cryptographic operation includes a step of concatenating the data unit parts. Thus the output data units of the cryptographic operations are restored on the basis of the parts of the output data units.

[0028] According to particular features, at least one cryptographic operation uses a look-up table for each data unit part, the input side of which receives said part of two permutated data units and provides said part of an output data unit.

[0029] Using look-up tables speeds up processing, because simply reading their outputs provides the result of the calculations, and also reduces vulnerability, because there are no intermediate steps that could risk giving information to an attacker.

[0030] According to particular features, at least one cryptographic operation applied to the permutated data units includes an exclusive-OR operation.

[0031] According to particular features, at least one cryptographic operation applied to the permutated data units is an encryption algorithm.

[0032] According to particular features, at least one cryptographic operation applied to the permutated data units is an AES encryption algorithm.

[0033] Thus, the present invention applies in particular to AES (Advanced Encryption Standard) coding and exploits its advantages, in particular in terms of consumption of resources, with modifications limited to those necessary to implement the randomized AES algorithm.

[0034] According to a second aspect, the present invention provides a device for cryptographic processing of secret data, that includes:

- permutation means adapted to permutate elementary data units of each data unit, and
- at least one operation means adapted to effect at least one cryptographic operation applied to the permutated data units,

characterized in that the permutation means are adapted to apply a different permutation to each of said data units, prior to said cryptographic operation.

[0035] Such a device can be a pocket electronic unit, such as a microcircuit card, for example one conforming to the ISO 7816 standard. Alternatively, it can be some other type of electronic unit, for example a computer (such as a personal

computer) or a USB key.

[0036] The method and device referred to above are typically implemented by a microprocessor executing instructions of a computer program. Thus, executing these instructions enables the microprocessor to process data stored in the device, for example in a random-access memory thereof. Other embodiments can nevertheless be envisaged, for example using an application-specific integrated circuit to execute the steps of the methods referred to above.

[0037] In these various contexts, the input data can be data received by the processing device from an external device, for example by means of a communication interface of the processing device. It can instead consist of data stored in the device (for example in non-volatile memory) or intermediate data obtained from result data of another transformation.

[0038] Similarly, at least some of the result data can be data to be sent to the output of the device, for example to be sent to the external device via the communication interface. The result data can nevertheless be only intermediate data, possibly used by the device in subsequent processes (for example calculation processes).

[0039] Each of the cryptographic algorithms mentioned above is used for example to provide at least part of an encryption, decryption, signature, cryptographic key exchange or cryptographic key generation operation.

[0040] In this type of application, the input data is, for example, a message (or part of a message) received from the external device and that is encrypted (or decrypted) in the processing device by means of the cryptographic algorithm or algorithms mentioned above and then forwarded via the communication interface at the output of the processing device.

[0041] According to a third aspect, the invention provides a computer program including instructions for executing the method of the present invention when said instructions are executed by a processor.

[0042] According to a fourth aspect, the present invention provides an information medium containing instructions for executing the method of the present invention when said instructions are executed by a processor.

[0043] The advantages, aims and particular features of this device, this computer program and this information medium being similar to those of the method of the present invention, they are not set out here.

[0044] Other particular advantages, aims and features of the present invention will emerge from the following description given, by way of nonlimiting explanation, with reference to the appended drawings, in which:

- figure 1 represents diagrammatically the main elements of one particular embodiment of a microcircuit card;
- figure 2 represents the general physical appearance of the microcircuit card illustrated in figure 1;
- figure 3 represents diagrammatically the prior art randomized AES algorithm;
- figure 4 represents, in flowchart form, steps for executing a first particular embodiment of the method of the present invention; and
- figure 5 represents, in flowchart form, steps for executing a second particular embodiment of the method of the present invention.

[0045] Figure 1 represents diagrammatically a data processing device 40 in which the present invention is implemented. This device 40 comprises a microprocessor 10 with which are associated on the one hand, for example by means of a bus 70, a random-access memory 60 and on the other hand, for example by means of a bus 50, a non-volatile memory 20 (for example of the EEPROM type).

[0046] The data processing device 40, to be precise its microprocessor 10, can exchange data with external devices via a communication interface 30.

[0047] There is represented diagrammatically in figure 1 the transmission of an input data unit X received from an external device (not shown) and forwarded from the communication interface 30 to the microprocessor 10. There is similarly shown the transmission of an output data unit S from the microprocessor 10 to an external device via the communication interface 30. This output data unit Y is the result of processing by the microprocessor 10 of data, generally the input data unit X, using secret data 80 internal to the system, for example a private key.

[0048] Although, for the purposes of illustration, the input data and the output data are shown on two different arrows, the physical means enabling communication between the microprocessor 10 and the interface 30 can take the form of a single unit, for example a serial communication port or a bus.

[0049] The microprocessor 10 is adapted to execute software (or a computer program) that enables the data processing device 40 to execute a method of the invention, examples of which are described with reference to figures 3 to 5. The software consists of a series of command instructions to the microprocessor 10 that are stored in the memory 20, for example.

[0050] Alternatively, the combination of the microprocessor 10, the non-volatile memory 20 and the random-access memory 60 can be replaced by an application-specific integrated circuit including means for executing the various steps of the data processing method.

[0051] Figure 2 represents a microcircuit card that constitutes one example of a data processing device of the invention as shown in figure 1. In this case the communication interface 30 is provided by the contacts of the microcircuit card. The microcircuit card incorporates a microprocessor 10, a random-access memory 60 and a non-volatile memory 20 as shown in figure 1.

[0052] This microcircuit card conforms for example to the ISO 7816 standard and is provided with a secure microcontroller that combines the microprocessor (or CPU) 20 and the random-access memory 60.

[0053] Alternatively, the data processing device can be a USB key, a paper document or a paper information medium incorporating in one of its sheets a microcircuit associated with contactless communication means. This device is preferably a portable or pocket electronic unit.

[0054] An example of the method of the invention applied to AES type symmetrical cryptographic processing during which the microprocessor 10 encrypts a message X to produce an encrypted message Y is described next with reference to figures 3 and 4.

[0055] Figure 3 shows diagrammatically the randomized AES algorithm that takes as input a block of 16 bytes of the message to be processed: $X = (x_i)_{i=0...15} \in GF(2^8)^{16}$.

[0056] The randomized AES algorithm executes a series of rounds, generally 10, 12 or 14 rounds, comprising the individual transformations represented in figure 3. The AES algorithm on which the randomized AES algorithm is based is a symmetrical encryption algorithm that takes as input a message X in the form of a block of 128 bits (16 bytes) and a key of 128, 192 or 256 bits.

[0057] The four principal steps of a round of the AES algorithm are:

- a non-linear (bit-by-bit addition) operation SubByte 110 of substitution of each byte x_i using an S-Box,
- a linear cyclic transposition operation ShiftRows 120,
- a linear matrix product operation MixColumns 130, and
- a linear operation AddRoundKey 140 of combining each byte with a round key.

[0058] Accordingly, the 16 input bytes are permutated in accordance with a predefined table during the transformation SubByte 110. These bytes are then placed in a 4x4 matrix and its rows subjected to a rightward translation by an increment that is variable as a function of the row during the transformation ShiftRows 120.

[0059] A linear transformation MixColumns 130 is then applied to the matrix in the form of binary multiplication of each element of the matrix with polynomials taken from an auxiliary matrix, this multiplication being subject to special rules according to the Galois group $GF(2^8)$, or finite body.

[0060] An intermediate matrix is then obtained by applying an XOR function to the matrix and another matrix linked to the round key.

[0061] These operations are repeated several times and define a "round". For a key of 128, 192 or 256 bits, AES respectively necessitates 10, 12 or 14 rounds.

[0062] The operations SubByte 110 and ShiftRows 120 can be carried out in the opposite order to that shown in figure 3 without impacting on the algorithm.

[0063] These steps are defined in the AES standard and are therefore not described in detail here except for what is relevant to the present invention.

[0064] In the randomized AES version, each of these steps is modified to take account of a random permutation. These various modified steps are defined in the paper by J.-S. Coron cited above.

[0065] As in standard masking, the countermeasure using permutation tables necessitates a random data string, obtained during a step 101, that is used at the beginning of the algorithm to generate a permutation P during a step 102. In the case of the encryption algorithm, during a step 104, the permutation step P is applied both to the message X to be encrypted and to the secret key K, to produce P(X) and P(K). These are permutated variables that are used by the encryption algorithm.

[0066] During subsequent steps of the encryption algorithm, the cryptographic operations are modified so that all the intermediate variables remain in the permutated form defined by the permutation P.

[0067] In the AES context, the intermediate variables are words of eight bits. The permutation P includes two permutations P_1 and P_2 of four bits and operates on a variable X of eight bits, or bytes, in accordance with the following formula, in which X_h and X_l respectively represent the most significant half-byte and the least significant half-byte of the byte X and "||" represents concatenation:

$$P(X) = P_2(X_h) \,||\, P_1(X_l).$$

[0068] Each time the algorithm is called, the permutations P_1 and P_2 are chosen at random in a set of permutations defined on $F_2{}^4$. In the remainder of the description it is assumed that the random variable P_1 (respectively P_2) associated with the random generation of p_1 (respectively p_2) satisfies the condition whereby the probability that the event $P_1 = p_1$ is equal to 2^{-4d} (respectively the probability that $P_2 = p_2$ is equal to 2^{-4d}), d being an integer (typically d = 4) for each value of p_1 (respectively p_2).

[0069] After the random permutation has been generated, it is applied to each byte of the message and the key. For each byte x, u = P(x) denotes the representation of x by the permutation P.

[0070] Each permutated value is processed by the AES round function, in which it is subjected to the above transformations 110, 120, 130 and 140. Each of the AES transformations must be rigorously implemented so that:

- the sensitive variables always appear in their form permutated by P, and
- the output of each transformation is in a permutated form.

[0071] Implementations of randomized AddRoundKey 140 and randomized MixColumns 130 are given below by way of example, it being understood that SubByte and ShiftRows can also be adapted to execute the present invention with the level of vulnerability reduced or even eliminated.

[0072] Randomized AddRoundKey 140 takes two bytes u = P(x) and v = P(y) as input and supplies as output P(x XOR y) where XOR represents the exclusive-OR operation. To this end, two tables XT_4^1 and XT_4^2 with inputs on 8 bits and outputs on 4 bits are defined for (u_l, v_l) and (u_h, v_h) in $(F_2^4)^2$:

$$XT_4^1(u_l \,||\, v_l) = p_1(p_1^{-1}(u_l) \text{ XOR } p_1^{-1}(v_l))$$

$$XT_4^2(u_h \,||\, v_h) = p_2(p_2^{-1}(u_h) \text{ XOR } p_2^{-1}(v_h)).$$

[0073] The tables XT_4^1 and XT_4^2 are calculated in parallel with P during a step 131 (see the description of Randomized MixColumns hereinafter) and stored in memory. An XOR function on eight bits XT_8 is then calculated during a step 145 using look-up tables (LUT) for u and v in F_2^8.

$$XT_8(u,v) = XT_4^2(u_h \,||\, v_h) \,||\, XT_4^1(u_l \,||\, v_l).$$

[0074] Randomized MixColumns 130 is calculated as a combination of multiplication-by-two operations, and XOR operations. To calculate Randomized MixColumns from $P(a_i)$, the representation of the bytes permutated by P, the XOR operations are calculated, using the function XT_8, during a step 132. A function D_2 is defined during a step 134 such that, when it is applied to u = P(x) during the step 136 of multiplication by two, $D_2(P(x)) = P(<\{02\} . x)$ is obtained, where {.} represents hexadecimal notation and "." represents multiplication modulo $x^4 + 1$. The permutated representation of the first byte of the output of Randomized MixColumns 130 is calculated as follows:

$P(a_0^{new}) = XT_8(D_2(a'_0), XT_8(D_2(a'_1), XT_8(a'_1, XT_8(a'_2, a'_3))))$ In the above formula a'_i represents the representation of a_i permutated by P. The other bytes in the outputs of randomized MixColumns are calculated in a similar way using XT_8 and D_2.

[0075] At the end of the last round, the inverse permutation P^{-1} is applied to each byte of the current result during a step 150 to reveal the encrypted message.

[0076] As shown in figure 4, the countermeasure of the present invention consists in a new implementation of the secure function XT_8. As in the prior art described above, two tables XT_4^1 and XT_4^2 going from eight bits to four bits are used, different from those described with reference to figure 3. However, before applying them to the half-bytes of data u(=P(x)) and v(=P(y)) to be combined using the XOR function, during preliminary steps 205 to 220, the representation P(y) of y permutated by P is exchanged for another, P*(y).

[0077] This change is effected by accessing the look-up table for the function P* o P^{-1}. Like the permutation P, the permutation P* is defined by two permutations p_1^* and p_2^* such that $P^*(x) = p_2^*(X_h) \,||\, p_1^*(x_l)$. For security reasons, p_1^* and p_2^* must be independent of p_1 and p_2, respectively.

[0078] The two new tables XT_4^1 and XT_4^2 are defined during a step 231 that replaces the step 131 described with reference to figure 3, as follows:

$$XT_4^1(u_l, v_l) = p_1(p_1^{-1}(u_l) \text{ XOR } p_1^{*-1}(v_l))$$

$$XT_4^2(u_h, v_h) = p_2(p_2^{-1}(u_h) \text{ XOR } p_2^{*-1}(v_h)).$$

[0079] Accordingly, the new XOR function XT_8 on eight bits is defined with reference to these tables such that, for u and v in F_2^8:

$$XT_8(u,v) = XT_4^2(u_h, v_h) \,||\, XT_4^1(u_l, v_l).$$

[0080] Consequently the following secure implementation of P(x XOR y) is used based on the permutated representations u = P(x) and v = P(y):

In a step 205, $v^* = P^* \text{ o } P^{-1}$ [v] is calculated.
In a step 210, $q_l = XT_4^1 (u_l, v_l^*)$ is processed.
In a step 215, $q_h = XT_4^2 (u_h, v_h^*)$ is processed.
In a step 220, $XT_8 (u, v) = q_h \,||\, q_l$ is calculated.

[0081] Since p_1 and p_2 are statistically independent permutations, it is possible to choose $P^* = p_1 \,||\, p_2$ without reducing the level of security against a first order SCA attack. In particular, p* can simply be defined on the basis of $P = p_2 \,||\, p_1$ by interchanging the roles of p_1 and p_2. With such a construction, the security method of the present invention requires no additional random permutation other than those already involved in the definition of the permutation P. Moreover, in this case, calculating $P^* \text{ o } P^{-1}$ [v] consists in processing $p_1 \text{ o } p_2^{-1}(V_h)$ and $p_2 \text{ o } p_1^{-1}(v_l)$. Compared to the implementation of XT_8 proposed in the prior art, the secure implementation of the present invention requires only one additional access to a look-up table and to two additional tables of 16 bytes that store the functions $p_2 \text{ o } p_1^{-1}$ and $p_1 \text{ o } p_2^{-1}$.

[0082] Moreover, these additional tables make it possible, without using additional memory, to optimize processing time and memory use in the randomized AES implementation described in the prior art document cited above.

[0083] This is because $p_2 \text{ o } p_1^{-1}$ and $p_1 \text{ o } p_2^{-1}$ are the tables p_{21} and p_{12} proposed in that document for calculating non-linear functions known as S-boxes in the function MIT (MIT standing for "Multiplicative Inverse Transformation"), to to achieve good time/memory tradeoff. Consequently, no additional memory space is necessary for conjointly implementing this particular embodiment of the present invention and this optimization.

[0084] The present invention is not limited to transformations of the AES symmetrical cryptographic algorithm, and can be applied to any type of algorithm, including DES and IDEA NXT (also known as FOX).

[0085] In a similar way to AES, FOX executes a number of rounds of operations, including a non-linear operation muX.

[0086] As shown in figure 5, in a second particular embodiment of the present invention, the method of cryptographic processing of data units u and v includes:

- a permutation step 320 during which permutation is effected of the elementary data units of each data unit u and v by applying different permutations to the elementary data units of u and the elementary data units of v that are preferably statistically independent, and
- at least one cryptographic operation 350 applied to the permutated data units.

[0087] Note that here the expression "elementary data units" primarily refers to binary data, i.e. bits, but can also represent n-tuples of bits, not necessarily consecutive in the data, for example pairs, triplets or quadruplets of bits, each data unit including a plurality of elementary data units.

[0088] In some embodiments, for example that shown in figure 4, the method further includes a step 305 of decomposing each of the data units into data unit parts, each data unit part including a plurality of elementary data units. In the example represented in figure 4, these data unit parts are most significant half-bytes and least significant half-bytes.

[0089] In this case, during the permutation step 320, it is preferable to effect a different permutation for the same parts of the data units u and v, for example p_1 and p_2 for the parts of u and p_2 and p_1 for the corresponding parts of v. Where the parts of v are concerned, this permutation is effected by means of $p_2 \text{ o } p_1^{-1}$ and $p_1 \text{ o } p_2^{-1}$. Accordingly, in the preferred embodiment described with reference to figure 4, the permutation step includes:

- a first step of permutation of the data units u and v during which a first permutation p_1 is applied to a first part of each of the data units u and v and a second permutation p_2 different from the first permutation p_1 is applied to a second part of each of the data units u and v, the first and second permutations being different and preferably statistically independent, and

- a second step of permutation of P(v) into v* during which the second permutation p_2 is applied after the inverse permutation p_1^{-1} of the first permutation to the first part of the data unit v and the first permutation p_1 is applied after the inverse permutation p_2^{-1} of the second permutation to the second part of the data unit v.

[0090] More generally, during the step 320, the permutation can include:

- a first permutation step during which the same permutation is applied to the different data units, and
- a second permutation step during which different permutations are applied to the different data units.

[0091] In some embodiments, at least one cryptographic operation 350 includes a step of concatenating the data unit parts.

[0092] In some embodiments, at least one cryptographic operation uses a look-up table (the two tables XT_4^1 and XT_4^2 in the embodiment shown in figure 4) for each data unit part, the input side of which receives said part of two permutated data units and provides said parts of an output data unit.

[0093] In some embodiments, in particular that shown in figure 4, at least one cryptographic operation applied to the permutated data units is an operation including an exclusive-OR (XOR) operation.

[0094] In some embodiments, in particular that shown in figure 4, at least one cryptographic operation applied to the permutated data units is an encryption algorithm.

[0095] In some embodiments, in particular that shown in figure 4, at least one cryptographic operation applied to the permutated data units is an AES encryption algorithm.

Claims

1. Method for cryptographic processing of data units, that includes:

   - a permutation step during which elementary data units of each data unit are permutated, and
   - at least one cryptographic operation applied to the permutated data units,

   characterized in that, prior to said cryptographic operation step, a different permutation is applied to each of the said data units.

2. Method according to claim 1, **characterized in that** it further includes a step of decomposition of each of said data units into data unit parts, each data unit part including a plurality of elementary data units, and, during the permutation step, different permutations are effected for the same parts of said data units.

3. Method according to either of claims 1 or 2, **characterized in that** the permutation step includes:

   - a first permutation step during which the same permutation is applied to the different data units, and
   - a second permutation step during which different permutations are applied to the different data units.

4. Method according to either of claims 2 or 3, **characterized in that** the permutation step includes:

   - a first step of permutation of the data units (u and v) during which a first permutation (p_1) is applied to a first part of each of the data units and a second permutation (p_2), different from the first permutation, is applied to a second part of each of the data units, the first and second permutations being different, and
   - a second permutation step during which the second permutation is applied after the inverse permutation (p_1^{-1}) of the first permutation to the first part of one of the data units (v) and the first permutation after the inverse permutation (p_2^{-1}) of the second permutation is applied to the second part of said data units.

5. Method according to any one of claims 1 to 4, **characterized in that** said permutations are statistically independent.

6. Method according to any one of claims 2, 4 or 5, **characterized in that** at least one cryptographic operation includes a step of concatenating the data unit parts.

7. Method according to any one of the claims 2 or 4 to 6, **characterized in that** at least one cryptographic operation uses a look-up table for each data unit part, the input side of which receives said part of two permutated data units and provides said part of an output data unit.

8. Method according to any one of claims 1 to 7, **characterized in that** at least one cryptographic operation applied to the permutated data units includes an exclusive-OR operation.

9. Method according to any one of claims 1 to 8, **characterized in that** at least one cryptographic operation applied to the permutated data units is an encryption algorithm.

10. Method according to any one of claims 1 to 9, **characterized in that** at least one cryptographic operation applied to the permutated data units is an AES encryption algorithm.

11. Device for cryptographic processing of secret data, that includes:

- permutation means adapted to permutate elementary data units of each data unit, and
- at least one operation means adapted to effect at least one cryptographic operation applied to the permutated data units,

characterized in that the permutation means are adapted to apply a different permutation to each of said data units, prior to said cryptographic operation.

12. Device according to claim 11, that constitutes a pocket electronic unit such as a microcircuit card.

13. Device according to claim 12, that conforms to the ISO 7816 standard.

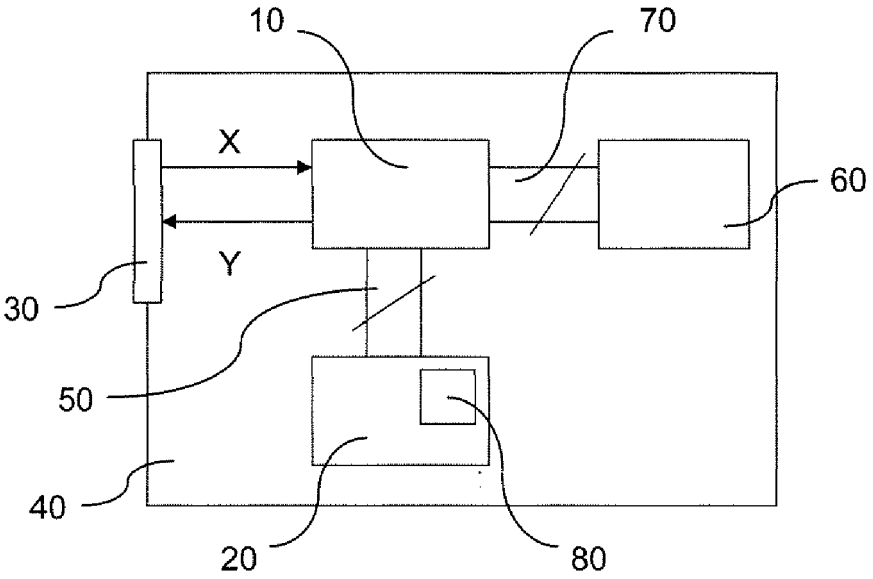

Figure 1

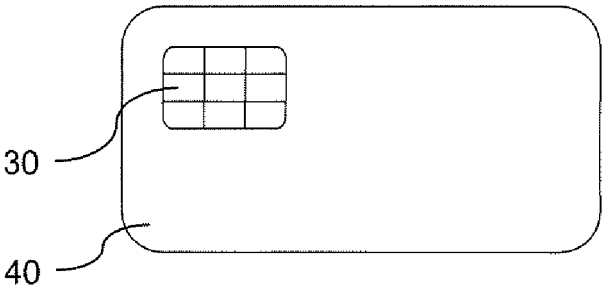

Figure 2

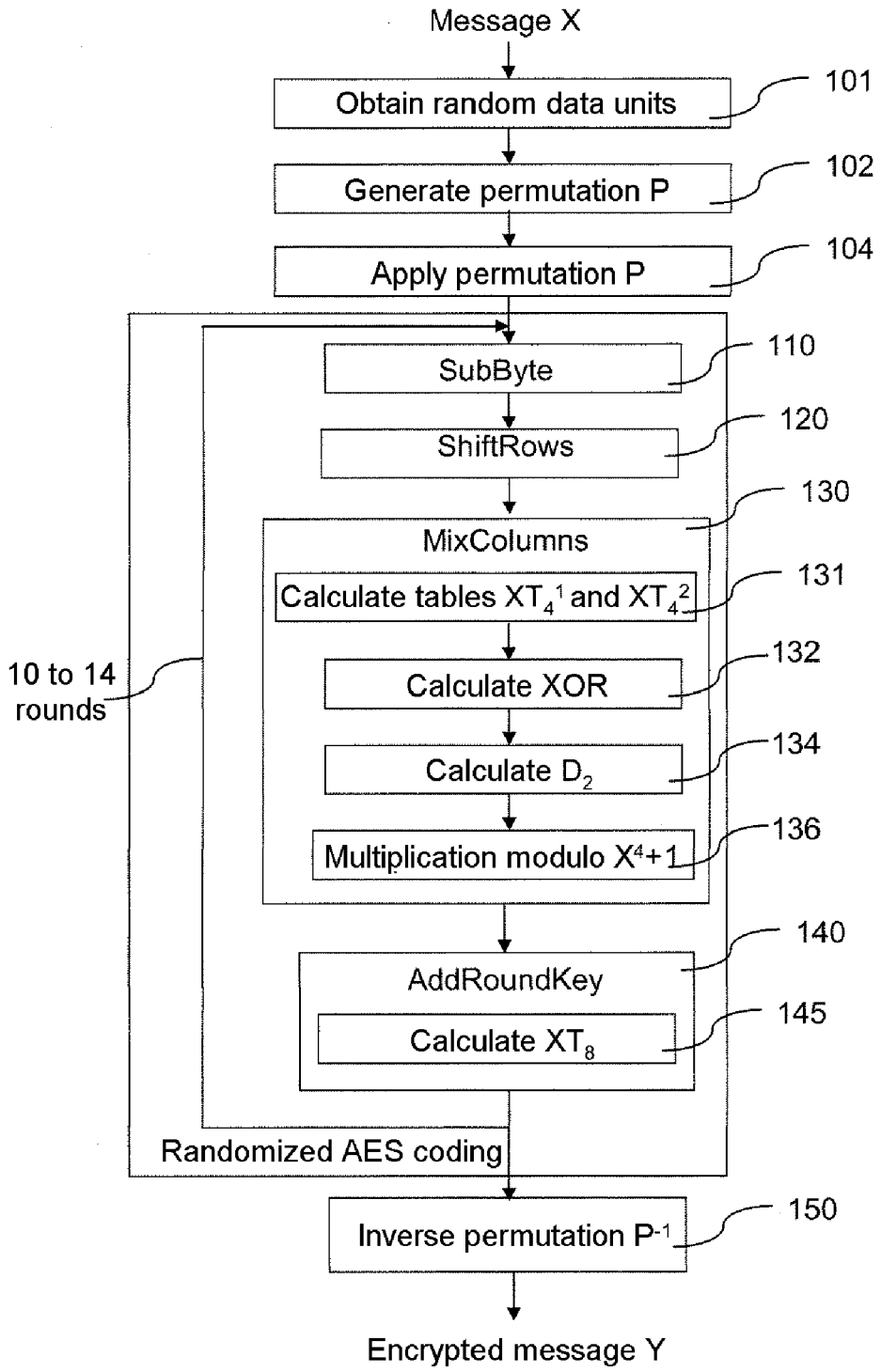
Message X
Obtain random data units
101
Generate permutation P
102
Apply permutation P
104
SubByte
110
ShiftRows
120
MixColumns
130
Calculate tables XT4^1 and XT4^2
131
Calculate XOR
132
Calculate D2
134
Multiplication modulo X4+1
136
AddRoundKey
140
Calculate XT8
145
10 to 14 rounds
Randomized AES coding
Inverse permutation P-1
150
Encrypted message Y

Figure 3

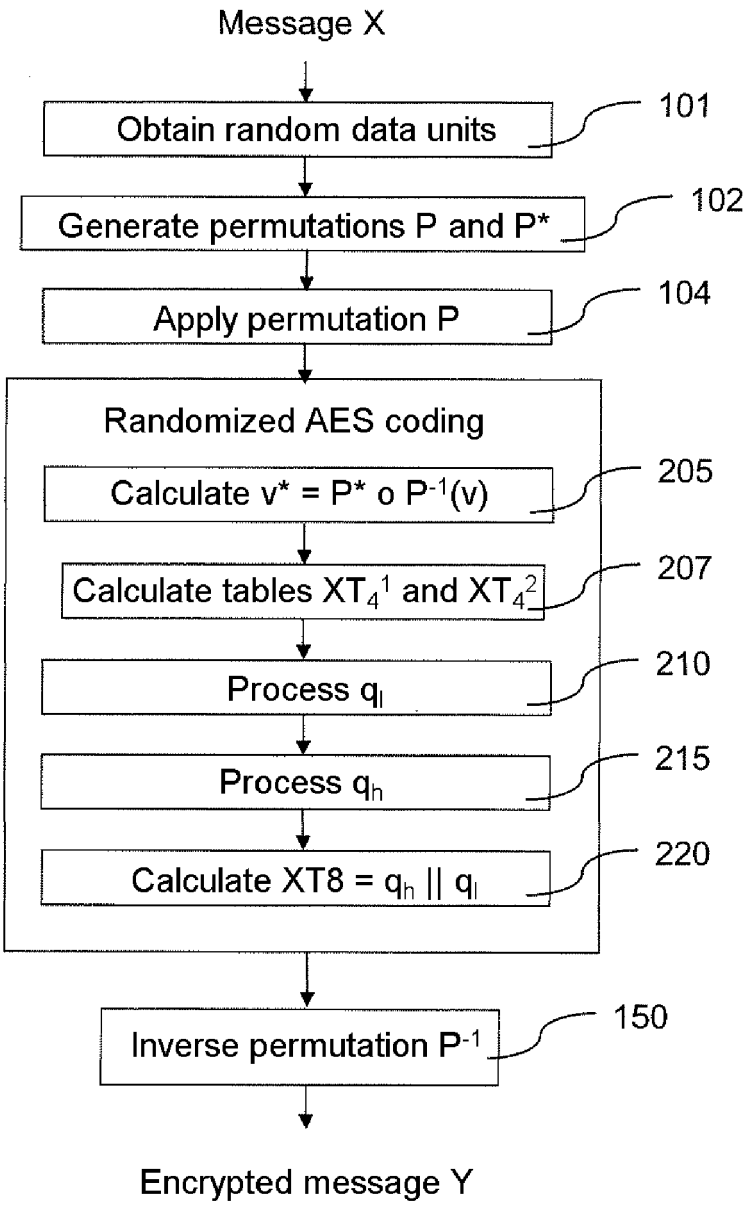
Message X
Obtain random data units
101
Generate permutations P and P*
102
Apply permutation P
104
Randomized AES coding
Calculate v* = P* o P-1(v)
205
Calculate tables XT4 1 and XT4 2
207
Process ql
210
Process qh
215
Calculate XT8 = qh || ql
220
Inverse permutation P-1
150
Encrypted message Y

Figure 4

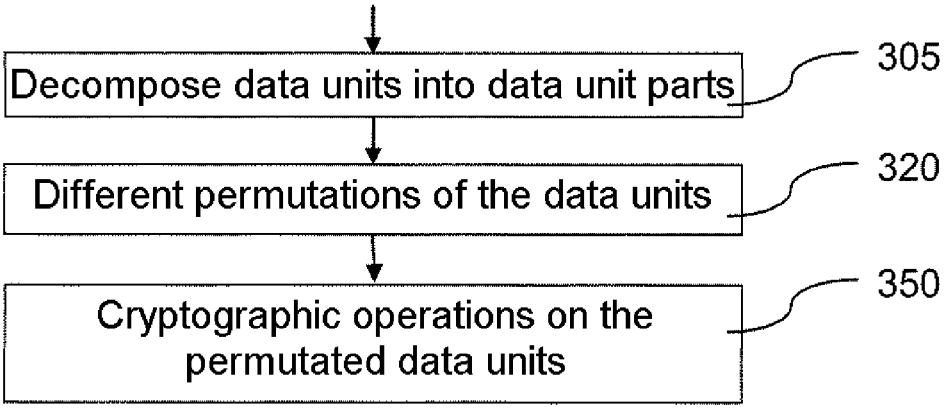
Decompose data units into data unit parts
305
Different permutations of the data units
320
Cryptographic operations on the permutated data units
350

Figure 5

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 10 16 5439

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
X	EMMANUEL PROUFF, ROBERT MCEVOY: "First-Order Side-Channel Attacks on the Permutation Tables Countermeasure", CRYPTOGRAPHIC HARDWARE AND EMBEDDED SYSTEMS - CHES 2009, SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, [Online] vol. 5747, 28 August 2009 (2009-08-28), pages 81-96, XP019128085, ISBN: 978-3-642-04137-2 Springer online Retrieved from the Internet: URL:http://www.springerlink.com/content/3476257h56458007/fulltext.pdf> [retrieved on 2010-06-16] * the whole document *	1-13	INV. H04L9/06
X	WO 02/03605 A1 (KONINKL PHILIPS ELECTRONICS NV [NL]) 10 January 2002 (2002-01-10) * abstract * * page 1, line 25 - page 2, line 5 * * page 2, line 33 - page 6, line 22; figures 1,2 *	1,2,5-13	TECHNICAL FIELDS SEARCHED (IPC) H04L
X	US 6 278 783 B1 (KOCHER PAUL C [US] ET AL) 21 August 2001 (2001-08-21) * abstract * * column 1, line 65 - column 3, line 27 * * column 6, line 20 - column 7, line 8 * * figure 1 *	1-13	

-/--

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	20 January 2011	Bec, Thierry

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

3

EPO FORM 1503 03.82 (P04C01)

Europäisches Patentamt
European Patent Office
Office européen des brevets

EUROPEAN SEARCH REPORT

Application Number
EP 10 16 5439

DOCUMENTS CONSIDERED TO BE RELEVANT

Category	Citation of document with indication, where appropriate, of relevant passages	Relevant to claim	CLASSIFICATION OF THE APPLICATION (IPC)
A,D	J.-S. CORON: "A New DPA Countermeasure Based on Permutation Tables", 10 September 2008 (2008-09-10), SECURITY AND CRYPTOGRAPHY FOR NETWORKS; [LECTURE NOTES IN COMPUTER SCIENCE], SPRINGER BERLIN HEIDELBERG, BERLIN, HEIDELBERG, PAGE(S) 278 - 292, XP019104377, ISBN: 978-3-540-85854-6 * the whole document * -----	1-13	

TECHNICAL FIELDS SEARCHED (IPC)

The present search report has been drawn up for all claims

Place of search	Date of completion of the search	Examiner
Munich	20 January 2011	Bec, Thierry

CATEGORY OF CITED DOCUMENTS

X : particularly relevant if taken alone
Y : particularly relevant if combined with another document of the same category
A : technological background
O : non-written disclosure
P : intermediate document

T : theory or principle underlying the invention
E : earlier patent document, but published on, or after the filing date
D : document cited in the application
L : document cited for other reasons

& : member of the same patent family, corresponding document

EPO FORM 1503 03.82 (P04C01)

3

ANNEX TO THE EUROPEAN SEARCH REPORT ON EUROPEAN PATENT APPLICATION NO. EP 10 16 5439

This annex lists the patent family members relating to the patent documents cited in the above-mentioned European search report. The members are as contained in the European Patent Office EDP file on
The European Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

20-01-2011

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
WO 0203605 A1	10-01-2002	AU 6908601 A	14-01-2002
		CN 1383648 A	04-12-2002
		EP 1303941 A1	23-04-2003
		JP 2004502965 T	29-01-2004
		TW 527811 B	11-04-2003
		US 2002027987 A1	07-03-2002
US 6278783 B1	21-08-2001	IL 139935 A	19-06-2005
		US 2001053220 A1	20-12-2001

EPO FORM P0459

For more details about this annex : see Official Journal of the European Patent Office, No. 12/82

REFERENCES CITED IN THE DESCRIPTION

This list of references cited by the applicant is for the reader's convenience only. It does not form part of the European patent document. Even though great care has been taken in compiling the references, errors or omissions cannot be excluded and the EPO disclaims all liability in this regard.

Non-patent literature cited in the description

- Towards Sound Approaches to Counteract Power-Analysis Attacks. **Suresh Chari ; Charantjit S. Jutla ; Josyula R. Rao ; Pankaj Rohatgi.** Proceedings of Advances in Cryptology - CRYPTO'99. Springer-Verlag, 1999, 398-412 **[0005]**
- A new DPA countermeasure based on permutation tables. **J.-S. Coron.** ''Security and cryptography for networks'', 6th international conference SCN 2008. Springer, 2008, vol. 5229, 278, 292 **[0009]**